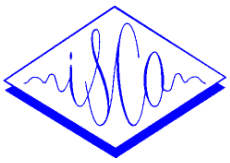

ITRW on Multi-Modal Dialogue
in Mobile Environments
Kloster Irsee, Germany
June 17-19, 2002

# MASTERING AGENT COMMUNICATION IN EMBASSI ON THE BASIS OF A FORMAL ONTOLOGY

Yves Forkl[*], Michael Hellenschmidt[**]
[*]*FORWISS, Haberstr. 2, 91058 Erlangen, Yves.Forkl@forwiss.de*

[**]*Fraunhofer-Institut für Graphische Datenverarbeitung, Fraunhoferstr. 5,*

*64283 Darmstadt, Michael.Hellenschmidt@igd.fhg.de*

**Abstract**: The development of agents, which are working together to accomplish complex tasks, needs efficient appliances – in particular if a great number of developers are involved in designing and implementing the functionalities of the agents and the communication between them. In this article it is shown how a formal ontology and an efficient monitoring of the communication can help to increase development efficiency and to decrease the needs of searching error causes. The ontology also provides ideal means for describing in a uniform and precise way the complete functionality that all of the various components of the system offer to the user.

**Key Words**: Ontology, Semantic Web, KQML, XML, Agent, Architecture, Multi-modal, Agent Platforms

# 1. SCOPE OF THE EMBASSI PROJECT

The aim of EMBASSI[1] is the development of multi-modally driven assistance for the usage of technical devices found in everyday life. The system is organised in a way to split the necessary functionalities (e.g. input / output-devices, dialogue management or assistants) into distributed components according to the generic EMBASSI architecture model given in figure 1 [Kirste et al. 2001].

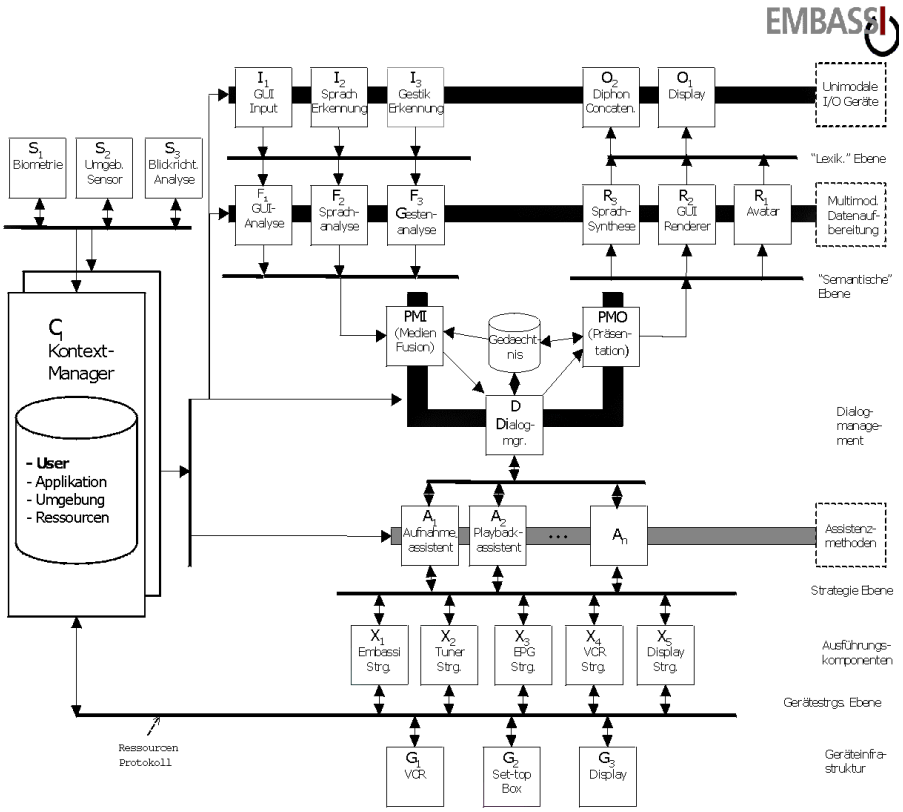


*Figure 1.* The generic EMBASSI architecture showing the components according to their position within the EMBASSI framework (in German). The router, coordinating the communication between all of these components, is left out of this figure.

[1]EMBASSI (``Elektronische Multimediale Bedien- und Service-Assistenz'', equivalent to ``Multimodal Assistance for Infotainment and Service Infrastructures''), is a joint project sponsored by the German Federal Ministry of Education and Research. See http://www.embassi.de/estart.html.

The EMBASSI project is engaged within three scenarios: The first scenario deals with the interactions of users with their home entertainment devices while the second scenario offers assistance within car environments. The last scenario should assist handicapped users in using terminal systems with the help of their personal EMMAs[2]. Internally, terminals and EMMAs are built up using the architecture given in figure 1.

By connecting the router of the EMMA to the router of terminals the EMMA components are able to use the infrastructure offered by terminal components. Terminal routers are therefore able to coordinate the communication between their own components and the communication between EMMA components and the internal components.

While the mechanisms of host detection, host connection and of the communication between two (or more) agent platforms can not be addressed in this paper, the following sections try to shed some light on the mechanisms of communication between the different components on the basis of a formal ontology encompassing the whole functionality that is available to the

## 2. THE USE OF AGENTS

As the generic EMBASSI architecture shows, each of the components is responsible for accomplishing specific tasks. For the communication between the different components we use agent technology. The FIPA Foundation (The Foundation for Intelligent Physical Agents) and the KQML (Knowledge Query and Manipulation Language) specification [FIPA 1997; Labrou, Finin 1997] define agents as the fundamental actors in a distributed system, offering several services to other agents or to the user. Agents are programs which can negotiate and can coordinate the transfer of information. Several principles of agent behaviour hold [Jennings 1997]:

Agents should be able to act autonomously and should perceive their environment as well as react on occurring changes (so-called events). We decided to implement an agent architecture with one server which behaves as a router. It offers two main services: the Transfer Service used to transmit messages from one agent to another (routing service), and the Information Service (White Pages service). In this set-up, agents have to communicate only with the router without needing to know where other agents reside (fig. 2).

[2] In EMBASSI, we name the Personal Digital Assistants “EMMA” (Electronic Multimedia Mobile Assistant).

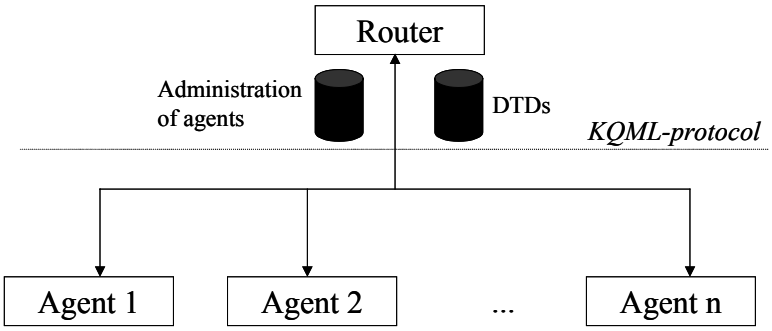


*Figure 2.* Agents communicate with each other using the router.

Therefore we decided to use the Knowledge Query and Manipulation Language (KQML) as Agent Communication Language (ACL) in the published version of 1997 [Labrou, Finin 1997]. Following KQML, an act of communication comprises a *performative* and several *parameters*. Performatives define which impact a message should have. We use the performatives *register* and *unregister* to make an agent known within the common agent-platform. After the agent has sent a message with the *connect* performative, it can be reached by other agents and the environment (e.g. the user) to communicate with. To send a question to another component[3] an agent uses the performative *ask-one*, in case it expects one answer, or *ask-all* in case it expects all possible answers. To reply to a question an agent uses the performative *tell*. With the performative called *achieve* a component issues a command to another agent. Parameters help to declare the sender and the receiver of a message or serve to organize messages. The Parameter *sender* specifies which agent initiated the communication. A reply-id (within the Parameters *reply-with* and *in-reply-to*) offers the possibility to refer to previous messages and provides self-organization.

The essential content of a message is located inside the *content* parameter; we chose XML as the (meta-) language for structuring the content. If an agent employs the *achieve* performative, it requests another agent to perform the content given in XML. Conversely, the receiving agent commits itself to try to make the message content given in XML to become true.

[3] The terms *component* and *agent* are used as synonyms within these sections.

Let us consider an example of a dialogue between two agents. If an agent XX wants to ask an agent YY for the actual time it sends the message: (ask-one :sender XX :receiver YY :reply-with 123 content (<time></time>)). Agent YY will answer with the message: (tell :sender YY :receiver XX :in-reply-to 123 :reply-with 456 :content (<time>2002-02-10 12:30:00</time>)).

While we have already stated that the message content is represented as an XML document, its concrete syntax and semantics have to be defined separately, in the form of a DTD that the content can be validated against. This means that the input and output of all components residing on the same data bus (or level, according to the EMBASSI architecture) is described using the same syntax and represented with consistent semantics.

# 3. A FORMAL ONTOLOGY FOR EMBASSI

## 3.1 Architectural Relevance of the Ontology

The exchange of "content", i.e. semantic information, between the EMBASSI agents undeniably requires the definition of a clear and unambiguous language allowing to convey meaning, cf. [Sowa 1999]. This is particularly true at the interface between dialogue processing and execution components: As opposed to conventional FSA-based monolithic dialogue-application systems with direct mapping from user utterances to system actions, the novel, highly modular EMBASSI architecture radically separates the dialogue manager from the application modules, as can be seen in figure 1. As a corollary, the semantics of the application functionality can no longer be hidden in a kind of black-box dialogue-and-application system, but must be made explicit. This explicit definition supplies the language to use in the messages exchanged between the components that interact strictly on the principle of the division of labour (cf. [Görz 2002]).

The dialogue manager (D in figure 1), in collaboration with the speech parser ($F_2$), identifies application-relevant user goals in multi-modal input received from the user and translates them into calls to assistance applications (A) which are in charge of trying to achieve these goals by executing the necessary tasks. In the case of an assistant module detecting that some information is missing, the module requests the dialogue manager to get this piece of information, e.g. by querying the user. Definitive success or failure of task execution is always fed back from the applications to the dialogue manager to enable it to keep its knowledge about the situation permanently up to date.

## 3.2 The Ontology as the System's Internal Lingua Franca

The various application functions and objects involved in the above-mentioned type of communication need to be precisely defined in order to allow for a compatible and consistent representation of the system's internal semantics in each message sent. This guarantees that all of the agents understand each other when they talk about their beliefs as well as that input from the user may be interpreted correctly as a user goal that provokes the desired system reaction. Furthermore, an unambiguous and logically correct representation of the application semantics is indispensable for the dialogue manager which can't do without reasoning continuously about the state of affairs concerning the identification and realization of user goals ([Ludwig 2001]).

The EMBASSI ontology – a hierarchy of concepts formalizing the semantics – furnishes the necessary language, which functions as a standardized system-wide terminology encompassing the whole world of functions and objects referred to by all applications and assistants for audio/video control at home and in the car.

## 3.3 Multi-Modality in the Ontology

Due to the multi-modal orientation of EMBASSI, its ontology has been designed specifically to better support the treatment of multi-modal input and output in the appropriate modules of the system which are called PMI and PMO (see figure 1; for details on these components see [Michelitsch et al. 2000]).

One main issue here is mode-independent interpretation of user input: e.g., saying "yes" or nodding one's head should invariably be recognized as an act of confirmation. This is assured by two principles: First, the fundamental separation of language and application modelling (see below) which means keeping mode-specific information out of the application area, and second, the persistent reduction of equivalent expressions originating from different modes to the same application concept structure to represent its semantics, making up a working principle of PMI and PMO.

Additionally, conceptual structures have been devised that deal with co-reference between different modes, as is illustrated by the formulation of a user goal by means of simultaneous speech and pointing, e.g. when the user points to a remote-controlled lamp and says "Switch that light on!". To enable processing of this co-reference, our ontology provides not only structures that allow to unify the speech- and pointing-specific representations of the device, but also mode-specific concepts that can

express, e.g., the generic action that the user seemingly wants to apply to the lamp he or she points to when simultaneously specifying the desired action by means of natural language.

## 3.4 Formulating the Ontology in Description Logics

Each of the functions and objects that the application modules of the system make available to the user must be both uniquely named and given a precise semantic definition in a formalism that is suitable for automate reasoning. We decided to represent our ontology using Description Logics[4], which are sublanguages of first order logic (see [Donini et al. 1996]). We have found them to be extremely useful for practical use, as they offer decidable, sound and complete algorithms for subsumption. Thus, we organized the conceptual knowledge about the internal semantics of the EMBASSI system in the form of descriptions of *concepts*, which are the basic units of a terminological hierarchy in DL, together with their properties, stated as *roles* (binary relations).

It is clear that the quality of the defined terminology depends on an optimal integration of each concept into the appropriate place in the ontology. This also holds for the concept's logical relations to other concepts, i.e. the role its instances may play in the identification of instances of another concept (a role thus compares to a feature or a slot).

## 3.5 Overall Structure of the Domain Model

The domain model tree defined by the ontology has three branches: First, the linguistic domain part which is dedicated to lexical semantics and linguistic units, hence language-specific. Second, the discourse domain part that is containing dialogue-related knowledge and describes dialogue objects (like utterances, dialogue goals etc.). Third and most important by size, the application domain part concerned with modelling the "world" of the applications and assistants, thus defining the language that allows the agents to talk about their functionality.

As already stated, this distinction reflects the neat architectural separation between, on the one hand, tasks that fall to the applications like developing a plan to satisfy an application-related user goal and, on the other hand, dialogue management tasks like identifying the user's communicative intention (speech act) and finding a way to query the user for missing information. If the linguistic modelling is delimited from these two parts, this is to ensure that complex language-dependent knowledge is kept out of

[4]http://dl.kr.org/

the application modelling proper, which is a prerequisite to optimal multi-modal input processing, as explained in detail above.

Using such a modular domain model also means that the domain model parts are easier to augment independently from each other and that multi-lingual modelling is easier to realize because it only affects the linguistic domain part.

The interconnection between the linguistic and application parts is particularly close, however: each application concept that the user might possibly talk about must be associated with a lexical concept, which corresponds to the series of synonymous words of the modelled language that are typically used to refer to the object or action represented by the application concept.

## 3.6 Characteristics of the Domain Model Parts

To provide for higher flexibility in extending our ontology and for conformity with standards, we chose the IEEE *Standard Upper Merged Ontology*[5] (cf. [Niles et al. 2001]) as the Generic Base Model upon which we built the discourse and application domain parts. This should allow for better integration of our ontology with current and future ontologies of various kinds, which is an highly important issue in the context of the Semantic Web where the fusion of incompatible ontologies has shown to be a major problem. The linguistic part, however, requires a different ontological basis that accounts for lexical-semantic relations rather than factual relations. This is why we organized our linguistic concepts according to the conceptual interlingua hierarchy defined by *EuroWordNet*[6] (cf. [Vossen 1999]), thereby facilitating multi-lingual extensions of the domain model: lexical concepts of different languages can in most cases be derived from the same base concepts, only sometimes requiring the introduction of new interlingua superconcepts.

When modelling the application knowledge, we first asked the developers of the application and assistant modules for OO interface definitions of their components. So we knew the complex data types as well as the functions (each with a set of parameters) that the applications use. Subsequently, we derived the application domain model for the EMBASSI domain immediately from these descriptions, integrating them into our ontology by linking the new concepts to those of our Generic Base Model. As the hierarchical and inheritance structures typical of OO languages are very much akin to those of ontologies, this approach facilitated the establishing of the ontology. In addition, conformance of the ontology with

[5] http://suo.ieee.org/
[6] http://www.hum.uva.nl/~ewn/

the data structures of the applications was much easier to achieve this way. (For a similar approach to ontology development by automatic exploitation of pre-existing formal descriptions cf. [Calvanese et al. 1998]).

## 3.7 Transformation of the DL Ontology into an XML DTD

Those parts of the EMBASSI ontology that are relevant to inter-agent communication, i.e. the "leaf concepts" of the terminology, have been transformed from the DL domain model into XML DTDs, as the XML 1.0 standard has been adopted for the encoding of message contents in EMBASSI. (In contrast, the inheritance structures of the ontology are neither necessary for the exchange of messages at the assistance level of the architecture – where essentially function calls are conveyed –, nor can hierarchical structures be represented in XML in a reasonable way without using XML Schema, RDFS or similar formalisms.) User-oriented EMBASSI agents like the dialogue manager and the assistant modules refer to a DTD called *sem.dtd* when communicating about instances of application concepts; verification of the syntax and semantics of the message content is possible by having the parser validate the content against the DTD.

# 4. FUNCTIONALITY OF THE ROUTER WITH RESPECT TO THE EMBASSI ONTOLOGY

To enforce that the agents use the correct syntax within their message content, the router is able to verify the content of a message by validating it against the DTD mentioned in the *ontology* parameter of the KQML message. This checking option can be switched on and off during system runtime (figure 3). So the router is able to reject a communication act if the sending agent uses a syntax that the receiver cannot understand. To give an example: Agents controlling video recorders would accept only messages like:

```
(achieve
 :sender xxx
 :receiver VCR-Agent
 :reply-with 123
 :ontology sem.dtd:Record
 :content (
        <Record>
          <AvEvent>?</AvEvent>
        </Record>)
```

)

The precise specification of *AvEvent* – the TV programme concept – and of the rest of the semantics descriptions laid down in our main DTD called *sem.dtd* is beyond the scope of this paper. (For further examples please refer to [Alexa et al. 2001].) If an agent keeps to the DTD which is defining this syntax, the router will pass the message through. Otherwise the router will deny the message and reply to the sender with an error message. Thus, the VCR agent does not have to bother with possibly getting incomprehensible messages: it can focus on its substantial tasks.

The router evaluates the *ontology* parameter to ascertain the name of the DTD and the root element. The name of the DTD and of the root element must be given within this parameter separated by a colon. The router combines this information and the message content to build a regular XML file which might look like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
   <!DOCTYPE Record SYSTEM "sem.dtd">
  <Record><AvEvent>?</AvEvent></Record>
```

This file containing a *doctype* declaration in its prologue is only used by the router, not by the individual agents, because validation only occurs in the router. In the case of successful validation, an agent only receives the XML content of the message without any direct reference to the DTD it conforms to (the receiver is, however, free to evaluate the ontology parameter of the KQML message by itself if it really cares about the document type used). Central DTD checking of message content has several advantages: First, the current version of the DTD *sem.dtd* used for validation can be kept in one location instead of having to be redistributed to each project partner who is maintaining a component. Second, only messages with valid content are passed on by the router: this not only keeps traffic low, but also reduces the need, on the side of the receiving agents, to check for syntactic errors when processing the input.

The router's internal XML file is checked for validity by running an XML parser. If the document is valid the graphical user interface of the router displays "true" for this communication act to the VCR-Agent in its DTD Checker window and the message is pulled through. In case it is not valid the graphical user interface shows "false" and the router sends an error message to the sender together with the error that was found while checking the XML document against its DTD.

Figure 3 shows an example of how the router handles different communication acts. On the right side one can see that the message the DM-

Agent wants to send to the VCR-Agent is valid (first row on the right side). Therefore the router pulls the message through to the VCR-Agent ("DM-Agent → VCR-Agent" on the first row on the left side). The following message is not valid (second row on the right side). The router reacts to this error by sending a message to the DM-Agent informing it about the error (second row on the left side).

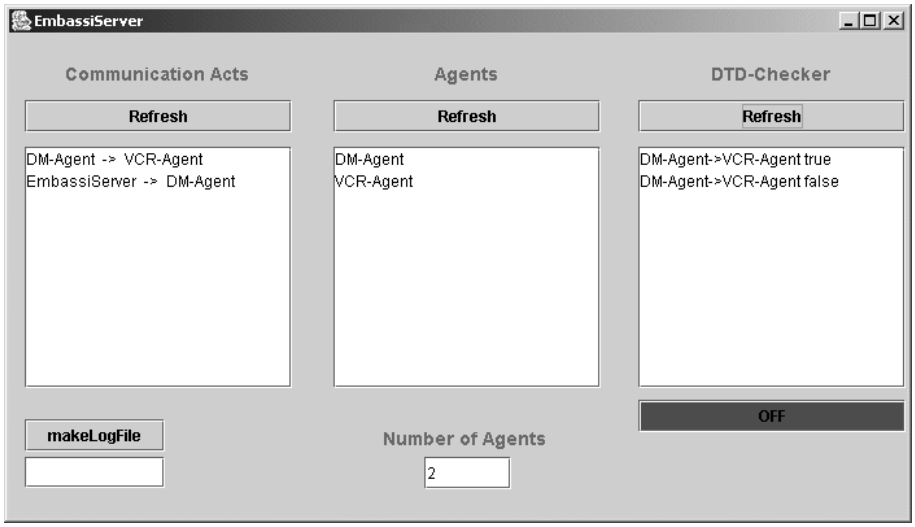


*Figure 3.* The graphical user interface of the router showing the Communication Acts on the left, the connected Agents in the middle and the results of the DTD checking on the right. The Button beneath can be used to switch DTD checking on and off.

Of course, checking the XML present in the content of every message costs time. Experiments with a router and several agents running on two Windows-2000 PCs yielded the following results: An average time for sending a message from one agent to another without DTD checking of 20 ms and an average time for sending a message from one agent to another employing the DTD-Checker of 400 ms.

Let us consider the advantages that, in our view, make it worth while investing this time.

## 5. ADVANTAGES

The main technical advantages of this approach are a reduction of communication acts in the case of syntax errors and a reduction of time and costs during the development of a complex agent-based architecture.

As shown in figure 1, the EMBASSI framework consists of many components, developed by a great variety of project partners. Each partner, contributing one or more agents, has to work together with other partners to get the communication between their components running. In order to test the communication, meetings in several cities had to be organized. Before the implementation of the DTD checking system the source of errors was hard to be determined. After integration of the DTD checking system in the testing, allowing to verify conformance to *sem.dtd*, the various reasons for errors were becoming obvious immediately.

If the router has passed a message through to the receiving agent, this means the content of the message is supposed to be in correct syntax so that the receiver has to understand it. Any errors that happen when the receiver processes its input are regarded as belonging to the responsibility of the latter (or rather of its developer).

If the router sends an error message to the sender, the sending agent must have used an XML syntax which is not valid and therefore the sender (and its developer) is responsible for the error.

Thus, unnecessary discussions are avoided and the developers within the EMBASSI project can focus upon the functionalities of their components. Consequently, the time needed for development decreases. Once the whole system is running the DTD-Checker can be switched off, reducing the average time needed for pulling messages through and speeding up most of the communication in the whole system.

Finally, it should be stressed that the ontology certainly plays a crucial role in the presented approach of mastering the communication in a large distributed agent-based system, but that its importance for the system as a whole goes far beyond that: The ontology can be seen as a common ground without which the numerous system components would hardly be able to communicate at all. If the ontology has been established in a well-defined logical formalism and allows to assign an unambiguous semantic interpretation to each message content exchanged by the agents, it might give the idea of a tuning fork that helps the orchestra of agents play together.

## REFERENCES

Alexa M., Berner U., Hellenschmidt M., Rieger T (2001).: An Animation System for User Interface Agents. *Proceedings WSCG 2001, The 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, 2001.*

Calvanese D., Lenzerini M., Nardi D. (1998): Description Logics for Conceptual Data Modeling, Logics for Databases and Information Systems, *J. Chomicki and G. Saake eds., Kluwer, 1998*.

Donini F.M., Lenzerini M., Nardi D., Schaerf A. (1996): Reasoning in Description Logics. In: Brewka, G. (ed.), Foundations of Knowledge Representation, pp. 191-236. *Stanford: CSLI Publications, 1996.*

Görz G. (2002): How Can Theory Contribute to the Construction of Scalable Speech Dialogue Systems? In: *Proceedings of the First International Workshop on Scalable Natural Language Understanding*, Heidelberg, 23-24 May 2002 [forthcoming].

Kirste Th., Herfet Th., Schnaider M. (2001): EMBASSI – Multimodal Assistance for Infotainment and Service Infrastructures, EC / NSF Workshop Universal on Accessibility of Ubiquitous Computing: Providing for the Elderly, 22-25 May 2001, Alcácer do Sal, Portugal.

Jennings N.R., Wooldridge M.J. (1997): Agent Technology, *Springer Verlag, 1997.*

Specification of the Foundation for intelligent physical Agents, FIPA*, http://www.fipa.org, 1997.*

Labrou Y., Finin T.(1997): A Proposal for a new KQML Specification, *Technical Report TR CS-93-03, 1997.*

Ludwig B. (2001): Dialogue Understanding in Dynamic Domains, Proceedings of the 5th *Workshop on Formal Semantics and Pragmatics of Dialogue (BIDIALOG 2001), Bielefeld 2001.*

Michelitsch G., Settele C., Hilt P., Torge S. und Rapp S. (2000): "EMBASSI: Overview of the Architecture for a Multimodal User Interface for Future Home Entertainment Systems", *Proceedings of the 10th Sony Research Forum, Tokio/Japan, 2000, pp. 156-161.*

Niles I., Pease A. (2001): *Toward a Standard Upper Ontology*, In: Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), http://projects.teknowledge.com/HPKB/Publications/FOIS.pdf

Sowa, J. F. (1999): Knowledge Representation: Logical, Philosophical, and Computational Foundations*. PWS Publishing Co., Boston, 1999.*

Vossen P. (ed.)(1999): EuroWordNet – General Document (EuroWordNet documents No. LE2-4003 and LE4-8328, Part A, Final), http://www.hum.uva.nl/~ewn/docs/GeneralDocPS.zip